Zadanie 3

Superciągiem ciągów X i Y nazywamy każdy taki ciąg Z, że zarówno X jak i Y są podciągami ciągu Z.

Ułóż algorytm, który dla danych ciągów X i Y znajduje ich najkrótszy superciąg.

(ang. *Shortest Common Supersequence*, SCS)

Algorytm

Problem można rozwiązać przy użyciu programowania dynamicznego. Niech $X = x_1x_2...x_n$ oraz $Y = y_1y_2...y_m$ będą ciągami wejściowymi o długościach odpowiednio $n = |X|$ i $m = |Y|$.

Obliczanie długości SCS

Definiujemy tablicę $\mathrm{dp}[i, j]$ jako długość najkrótszego wspólnego superciągu dla prefiksów $X[1..i]$ oraz $Y[1..j]$. Wartości w tablicy obliczamy zgodnie z następującą rekurencją, zakładając indeksowanie od 1:

$$\mathrm{dp}[i,j] = \begin{cases} j & \text{dla } i = 0 \\ i & \text{dla } j = 0 \\ \mathrm{dp}[i-1, j-1] + 1 & \text{jeśli } x_i = y_j \\ \min(\mathrm{dp}[i-1, j], \mathrm{dp}[i, j-1]) + 1 & \text{jeśli } x_i \neq x_j \end{cases}$$

Ostateczna długość najkróteszego superciągu dla całych ciągów X i Y znajduje się w komórce $\mathrm{dp}[n, m]$.

Odtworzenie superciągu

Aby odtworzyć sam superciąg (a nie tylko jego długość), należy prześledzić ścieżkę w tablicy dp od komórki (n, m) do $(0, 0)$. Zaczynając od $i = n$, $j = m$, wykonujemy następujące kroki, dopóki $i > 0$ lub $j > 0$.

1. Jeśli $\boldsymbol{x_i = y_j}$: dodajemy x_i na początek wyniku i przechodzimy do komórki $(i-1, j-1)$.
2. Jeśli $\boldsymbol{x_i \neq y_j}$:
   i. Jeśli $\mathbf{dp[i-1, j] \leq dp[i, j-1]}$: dodajemy X_i na początek wyniku i przechodzimy do komórki $(i-1, j)$.
   ii. Jeśli $\mathbf{dp[i-1, j] > dp[i, j-1]}$: dodajemy Y_j na początek wyniku i przechodzimy do komórki $(i, j-1)$.
3. Jeśli $\boldsymbol{i = 0}$: dodajemy pozostały prefiks $Y[1..j]$ na początek wyniku i kończymy.
4. Jeśli $\boldsymbol{j = 0}$: dodajemy pozostały prefiks $X[1..i]$ na początek wyniku i kończymy.

Opracował: Filip Figzał, 2025

Dowód poprawności

W każdej komórce $\mathrm{dp}[i, j]$ przechowujemy długość najkrótszego wspólnego superciągu dla prefiksów $X[1..i]$ i $Y[1..j]$. Poprawność algorytmu wynika z zasady optymalności.

1. **Przypadki bazowe**:
   i. $\mathbf{dp[i, 0] = i}$:
   Najkrótszy superciąg dla $X[1..i]$ i pustego ciągu to sam $X[1..i]$, który ma długość i.
   ii. $\mathbf{dp[0, j] = j}$:
   Analogicznie dla $Y[1..j]$ i pustego ciągu.
2. **Krok rekurencyjny**:
   Rozważmy problem dla $\mathrm{dp}[i, j]$.
   i. Jeśli $\boldsymbol{x_i = y_j}$:
   Ostatni znak obu prefiksów jest taki sam. Możemy go użyć jako ostatniego znaku wspólnego superciągu. Pozostała część superciągu musi być najkrótszym superciągiem dla prefiksów $X[1..i-1]$ i $Y[1..j-1]$. Jego długość to $\mathrm{dp}[i-1, j-1]$. Zatem całkowita długość wynosi $\mathrm{dp}[i-1, j-1] + 1$.
   ii. Jeśli $\boldsymbol{x_i \neq y_j}$:
   Ostatnio znak superciągu musi być albo x_i, albo y_j.
      a. Jeśli wybierzemy x_i, musimy znaleźć SCS dla $X[1..i-1]$ i $Y[1..j]$, a następnie dołączyć x_i. Długość wyniesie $\mathrm{dp}[i-1, j] + 1$.
      b. Jeśli wybierzemy y_j, musimy znaleźć SCS dla $X[1..1]$ i $Y[1..j-1]$, a następnie dołączyć y_j. Długość wyniesie $\mathrm{dp}[i, j-1] + 1$.

   Wybieramy opcję, kóra dalej krótszy wynik, stąd $\min(\mathrm{dp}[i-1, j], \mathrm{dp}[i, j-1]) + 1$.

Ponieważ w każdym kroku podejmujemy optymalną lokalnie decyzję, która prowadzi do globalnego optimum, algorytm jest poprawny.

Analiza złożoności

Niech $n = |X|$ oraz $m = |Y|$.

- **Złożoność czasowa:** Algorytm wypełnia tablicę o wymiarach $(n+1) \times (m+1)$. Obliczenie wartości każdej komórki zajmuje stały czas $O(1)$. Odtworzenie ciągu trwa $O(n+m)$, ponieważ przechodzimy od (n, m) do $(0, 0)$. Zatem całkowita złożoność czasowa wynosi $O(n \cdot m)$.
- **Złożoność pamięciowa:** Wymagane jest przechowywanie całej tablicy dp, co zajmuje $O(n \cdot m)$ pamięci.